A triangular grid generation and optimization framework for the design of free-form gridshells

Qi-sheng Wang[1], Jun Ye[2,*], Hui Wu[3], Bo-qing Gao[1], Paul Shepherd[4]

1 College of Civil Engineering and Architecture, Zhejiang University, Hangzhou, China

2 Department of Civil and Environmental Engineering, Imperial College London, London, UK

3 Public Administration College, Zhejiang University of Finance & Economics, Hangzhou, China

4 Department of Architecture and Civil Engineering, University of Bath, Bath, UK

*Corresponding author: j.ye13@imperial.ac.uk

Abstract: Gridshells have been widely used in various public buildings, and many of them are defined over complex free-form surfaces with complex boundaries. This emphasizes the importance of general grid generation and optimization methods in the initial design stage to achieve visually sound and easy-to-manufacture structure. In this paper, a framework is presented to generate uniform, well-shaped and fluency triangular grids for structural design over free-form surfaces, especially those with complex boundaries. The framework employs force-based algorithms and a connectivity-regularization algorithm to optimize grid quality. First, an appropriate distribution of internal points is randomly generated on the surface. Secondly, a bubble-packing method is employed to increase the uniformity of the initial point distribution, and the points are connected using Delaunay-based triangularization to produce an initial grid with rods of balanced length. Thirdly, the grid connectivity is optimized using a range of edge-operations including edge- flip, collapse and split. The optimization process features a grid relaxation objective which includes the degree of the vertices, leading to improved regularity. As a final step, the grid is relaxed to improve fluency using a net-like method. As part of its contribution, this paper, therefore, proposes a metric for fluency, which can be used to quantitatively evaluate the suitability of a given grid for architectural and structural expression. Two case-study examples are presented to demonstrate the effective execution of the grid generation and optimization framework. It is shown that by using the proposed framework, the fluency index of the grid can be improved by up to 157%.

Keywords: gridshell; free-form surface; grid generation; dynamic relaxation; grid quality

Notation

i,j,k	serial number
n	total number of grid nodes or bubbles
k_n,k_Q	coefficients
k_b, k_c	linear spring constants
D_i	original diameter of i-th bubble
$\vec{d}_{ij}$	displacement vector from i-th bubble center to j-th bubble center
$\vec{T}_{ij}$	inter-bubble force between i-th bubble and j-th bubble
$\vec{T}_i$	resultant inter-bubble force at i-th bubble
$\vec{d}_{si}$	displacement vector from the center of i-th bubble to its closest point over the surface
$\vec{P}_{si}$	vector of force to attract i-th bubble to the surface
f_i	resistance force on i-th bubble
k_f	damping coefficient

m	lumped mass for each bubble
$\vec{v}_i$	velocity vector of *i*-th bubble
$\vec{F}_i$	resultant force acting on *i*-th bubble
$\vec{a}_i$	acceleration of *i*-th bubble
$\vec{s}_i$	current position of *i*-th bubble
d_i	degree of *i*-th vertex in a triangulation
n_i	virtual degree of *i*-th vertex
d_i^*	adjusted degree for *i*-th vertex
γ_i	weight of *i*-th vertex
λ	weight of interior vertices
$R(G)$	nodal degree residual of the grid G
$\vec{l}_{ij}$	vector of displacement from the *i*-th particle p_i to the *j*-th particle p_j
n_i	number of vertices connected to *i*-th vertex, that is, d_i
$\vec{l}_{ij}$	displacement vector from *i*-th particle p_i to *j*-th particle p_j , that is, $\vec{d}_{ij}$
l_0	constant slack-length of the spring
$\vec{T}_i'$	resultant spring force at *i*-th particle
n_e	total number of edges
$\vec{P}_{ci}$	boundary attractive force at *i*-th particle
q	triangular shape index
n_{int}	number of interior vertices
n_{bou}	number of boundary vertices
n^*	adjusted number of grid vertices
μ	the average nodal degree residual
δ_i	angle disfluency index of *i*-th vertex
Q	fluency index of a grid
ρ	proportion of the interior regular vertices
n_{inr}	number of interior regular vertices

1 Introduction

Gridshells as long-span roof structures are often the most striking part of a building from a designer's perspective, providing a sense of simplicity and elegance of appearance. The important features of gridshells are their appeal of uninterrupted spans, the smoothness of their continuum surface, the lightness of their grid cells, curve fluidity and, most importantly, their high structural efficiency, that can resist external actions through membrane stiffness [1]. Gridshells with planar, cylindrical, spherical and parabolic shapes have been widely used in practice [2-4], where engineers often use analytical equations to generate the nodal positions and member connectivities for structural design.

In recent years, parametric modeling and programming techniques in computer-aided design have allowed a new level of complexity in 3D free-form structural (surface) design, allowing architects more freedom to create inspiring

forms and to restructure their workflow. The aesthetically pleasing nature of gridshells attracts the attention of high-profile clients, and the number of new and complex free-form grid structures is increasing. Some buildings with complex shapes have recently been erected successfully, as shown in Fig.1.

(a) Yas Viceroy Abu Dhabi Hotel, Abu Dhabi, UAE (b) Shenzhen Bay Sports Center, Shenzhen, China

Fig.1 Examples of free-form grid structures

Previous studies on free-form grid structures have concentrated on the structural design aspects, form-finding methodologies [5] and optimization techniques [6], and the associated research on grid generating methodologies is relatively scarce. However, it is not always easy to create an efficient grid structure on a given surface, despite grid generation being a key stage in the design of free-form grid shells. Gridshells are often highly efficient in terms of material use, therefore, in this paper efforts are made to generate grids with balanced rod lengths over a continuum surface, whilst the efficiency of the material use and construction cost are assumed to be optimized subsequently in the structural design process. Grids generated should approximate the given surface as closely as possible. Grid members should all be of similar lengths to ease manufacture, connections, and equalize their buckling strengths, and each continuous member should pass over the surface in a fluid manner and avoid singularities of curvature (kinks), as shown in Fig.2.

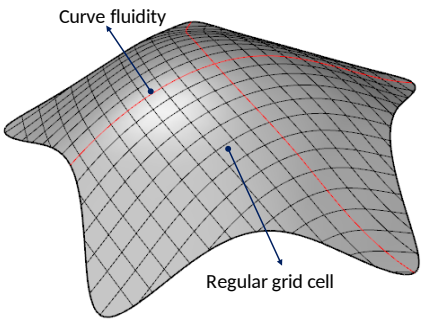


Fig.2 A gridshell with curve fluidity and regular grid cells

Unstructured grid generation methods have been developed for finite-element analysis (FEA), with the most prolific

methods being Advancing Front Technique[7], Mapping Method [8] and Delaunay Triangulation [9, 10]. For the interested reader, an excellent review of unstructured grid generation algorithms has been presented by Owen [11]. In graphics applications, many remeshing methods [12] have been proposed to improve the mesh quality in terms of vertex sampling, size grading and element shapes amongst other indexes. For example, Kammoun et al. [13] proposed an algorithm to remesh an available surface grid into an adaptive semi-regular mesh based on Voronoi tessellation and regular subdivision. The meshing methods singly pursue grid regularity by ignoring other design requirements of the grid, such as fluency and uniformity. And whilst they are not particularly focused towards the design of architectural grids, they do however provide an important underpinning methodology to improve grid regularity. Gridshells come in many forms, and are generally composed of triangular, quadrilateral or hexagonal grid cells. Su et al. [12] proposed an improved advancing front technique to generate triangular grids and the principle stress trajectories were used to determine edge directions. Gao et al. [14] developed a "guide line method" to generate triangular or quadrilateral grids with rods of balanced length and fluent lines. Peng et al. [15] presented a framework to generate meshes composed of a hybrid of both triangles and quads and the generated meshes fit the surface boundaries and curvatures.An excellent review of designing grid structures that consist of polyhedral cells has been presented by Pottmann et al [16].

This paper focuses on triangular grid generation and optimisation over a free-form surface, since triangulated grids are structurally efficient and are most widely used. For the triangulation of a given design domain, a point list is often generated first, and the points are subsequently connected to form triangles, often using Delaunay triangularization [9, 10]. With the topology of a grid determined, force-based methods have been used effectively to smooth (adjust the point positions) the grid. Force-based methods define proximity-based, repulsive/attractive inter-nodal forces between node-pairs and then carry out an explicit dynamic simulation with damping to achieve a force-balanced configuration of nodes. Diverse grid types and element sizes can therefore be achieved by adjusting the directions and magnitudes of the nodal forces. Shimada and Gossard [17] proposed a bubble-packing method for the grid generation on non-manifold geometry. They used the list of points spanning the domain to define the centres of packed bubbles, and the bubble locations were then optimised by iteratively solving equations of motion. They later extended the bubble-packing method by generating grids on trimmed parametric surfaces for finite element modelling [18]. Zheleznyakova et al. [19, 20] also proposed an approach to generate grids for finite element applications, but based on the concept of molecular dynamic modelling. Nodes were taken as likewise charged interacting units that could move to optimal positions in a parametric design domain of the NURBS surface. These particles were then linked into well-shaped triangles using Delaunay Triangulation algorithm. The resulted triangles were then mapped

from the parametric space to the physical space. However, this mapping of triangles from the parametric domain to the 3D surface causes metric distortions in the grids, limiting the method's application. Despite such physically-based methods, there is still a need to develop suitable methodologies to generate grids specifically for the structural design on free-form lattice shells that considers both grid uniformity and regularity.

This paper presents a framework for generating triangular grids with improved quality, employing a force-based smoothing technique and a connectivity regularization algorithm to optimize the grid topology. The framework starts with distributing a list of points spanning the design domain and applies a bubble-packing method and a Delaunay-like triangulation algorithm to uniformize point distribution and generate a uniform grid respectively. The connectivity of the uniform grid is then improved by iterative edge-operations to obtain a semi-regular grid with a small number of extraordinary nodes. The semi-regular grid is smoothed by a net-like method and the final gird is acquired. Section 2 introduces the NURBS technique and the gird to represent free-form surfaces. Section 3 gives an overview of our grid generation framework. In Section 4, a bubble-packing method is introduced with the dynamic simulation technique to achieve more uniformly distributed points, while a technique based on Delaunay triangulation is described to connect the points into a triangular grid. Section 5 then presents a series of geometry operations to improve the regularity of the grid connectivity. In Section 6, a net-like method is introduced to smooth the grid. In Section 7, metrics are defined to quantify the element and fluency of grids for gridshell applications. Two case studies are presented in Section 8 to illustrate the effectiveness of the framework and the resultant grids are evaluated by edge lengths, shape quality indexes and fluency indexes. This paper, therefore, uses two physical-based methods for the grid generation of free-form gridshells, a bubble-packing method and a net-like method.

2 Representation of surfaces

Although many mathematical models for the representation of surfaces have been proposed, NURBS (Non-Uniform Rational B-Splines) [21] tend to be the most common in engineering applications and are widely used due to their mathematically accurate illustrations of a variety of shapes, from a simple 2D line to a complex 3D surface, using a relatively small amount of information. This section presents a brief on the definition of NURBS to set the scene for the rest of the paper, since the proposed framework uses the NURBS models to parameterize the surfaces and curves in the grid generation process.

A NURBS surface represents an arbitrary geometric shape by adjusting its knot weights and control points and establishes a mapping relationship between the surface and its parametric domain. A complex CAD model is usually

represented by a multiple surface in the design of free-form gridshells. The multiple surface comprises a set of NURBS surface patches. Fig.3(a) illustrates a multiple surface made from five surface patches.

Discretized forms of surfaces (meshes/grids) are composed of a number of connected vertices, edges, and faces, as shown in Fig.3(b). An edge (the connection between a pair of vertices) forming only one face is defined as a boundary edge, whilst an edge which defines two faces is an interior edge. The endpoints of boundary edges are boundary vertices, whilst other endpoints are all interior vertices.

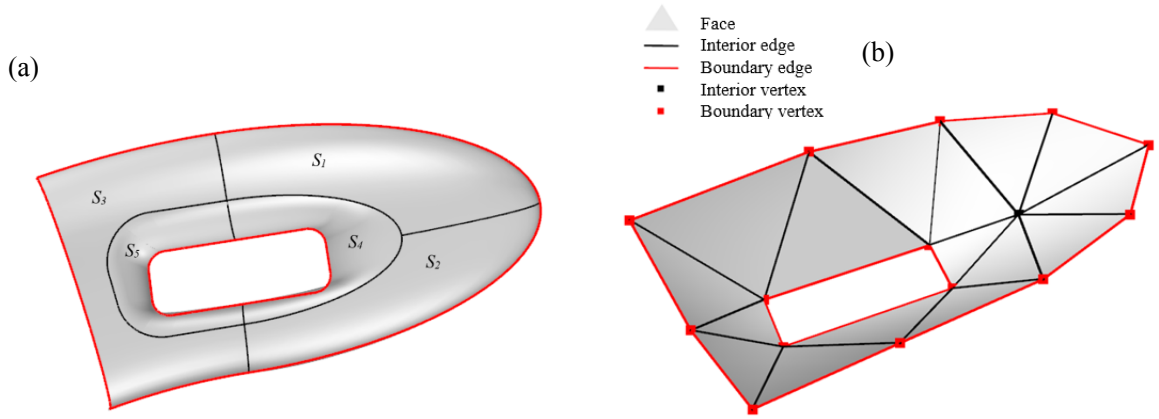


Fig. 3 A multiple surface and its grid representation

3 Overview of the framework

An overview of the grid generation framework is presented in Fig.4. The starting point is a surface geometry which remains fixed throughout the whole procedure. On this surface we wish to create an optimal grid shell

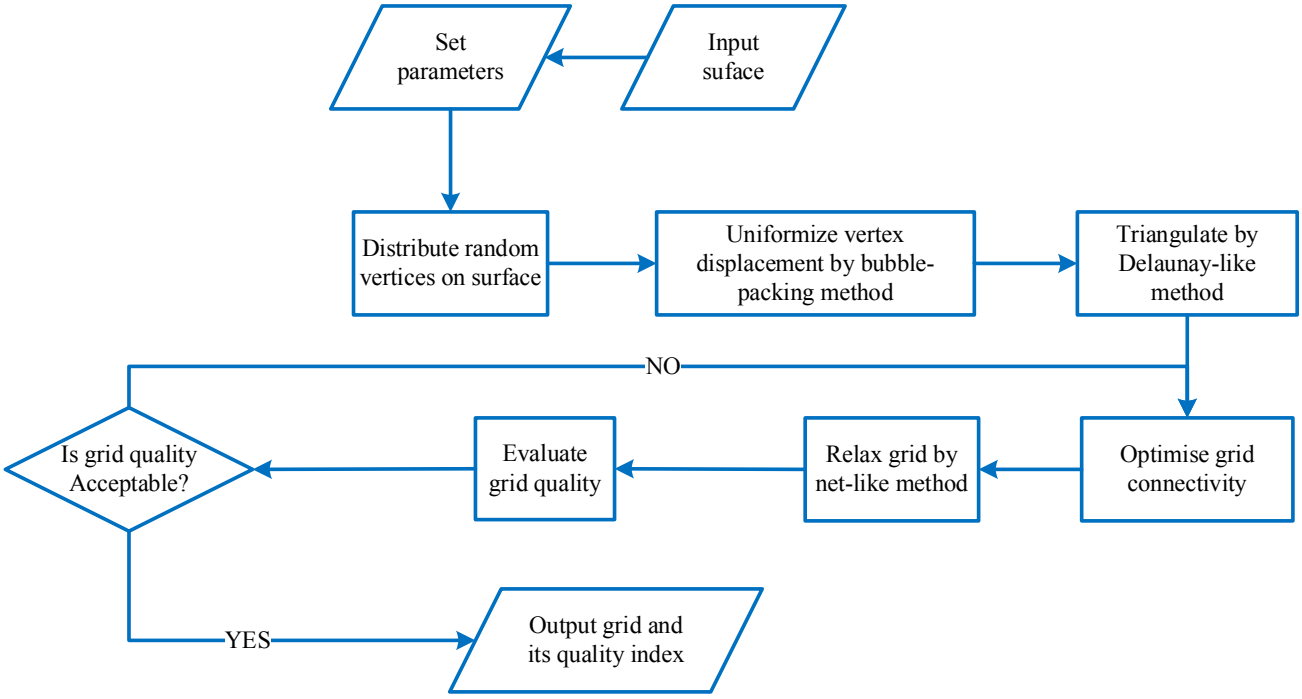


Fig.4 Flow diagram of our scheme

The steps needed to generate a high-quality grid using this framework are as follows:

(1) The bubble-packing method is used to uniformly distribute the randomly generated node distribution on a given surface by solving a series of dynamic motion equations to achieve a static equilibrium state.

(2) Delaunay triangulation-based method used to connect vertices into a triangular grid on the free-form surface.

(3) Connectivity of the grid is optimized based on the application of three basic edge operations: flips, collapses and splits.

(4) The net-like method smooths the grid on the surface without changing its topology.

(5) Grid quantitative measures to evaluate the grid, including edge-length index, grid shape quality and grid fluency index to allow comparison between initial and final designs.

The algorithms have been programmed in a plugin for the Rhinoceros-based parametric geometric modelling tool Grasshopper [22], providing a parametric modelling environment suitable for use in design. The detailed procedure of the grid generation framework would be presented in the following 3 sections using an example for illustration purposes.

4 Initial grid generation

To illustrate the grid generation and optimization framework proposed in this paper, a glass facade of an air-corridor structure from practice has been used as an example for the grid generation process, as shown in Fig.5. The gridshell

was due to be built as a facade of a large-span space structure and is a landmark in Wenzhou, China. The facade is 650*m* in length, 240*m* wide and 33*m* in height. Its surface model, *S*, is formed of 11 NURBS surface patches.

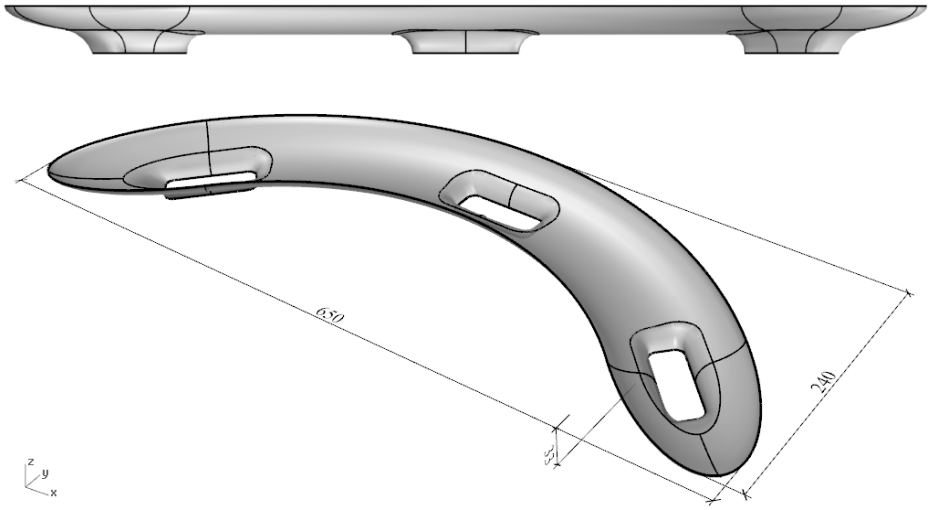


Fig.5 Surface representing a facade of a large-span space structure (units: *m*)

A set of points is first randomly generated on the surface, with the number of points, *n*, given by Eq. (1) based on the area *A* of the surface and a desired triangle area A_0 or edge length l_0 (generally estimated by the designer from experience):

$$n = k_{\mathrm{n}}\frac{A}{2A_0} = k_{\mathrm{n}}\frac{A}{2\times\frac{\sqrt{3}}{4}l_0^2} = \frac{2}{\sqrt{3}}k_{\mathrm{n}}\frac{A}{l_0^2} \tag{1}$$

The coefficient of k_{n} is initially presumed to be 1.0, however, it can be adjusted by trial and error in an interactive manner. If the random placement of points leads to an element in the final grid being smaller than the desired l_0, k_{n} can be decreased to reduce the number of points and obtain longer members. As Fig.6 shows, A_0 is the area of an equilateral triangle with side length l_0. For each vertex of a triangle, the area allocated to the vertex will be $\frac{1}{3}A_0$. Since each vertex has 6 adjacent triangles, the total area allocated to each vertex will be $\frac{1}{3}A_0 \times 6 = 2A_0$.

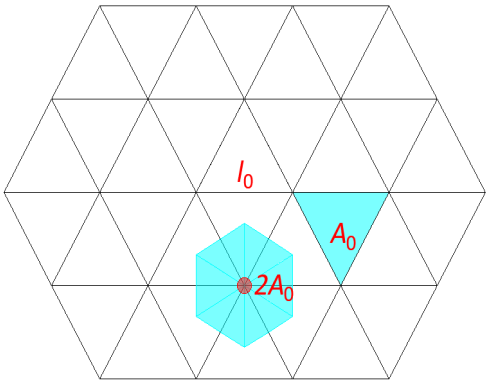


Fig.6 Allocated area for an individual vertex ($A_0 = \frac{\sqrt{3}}{4} l_0^2$ is the area of an equilateral triangle with side length l_0)

Since the initially generated points are randomly distributed in the domain, the potential connectivity between points will not be obvious (as shown in Fig.7) and a method to control the spacing within the domain is also needed to produce grids with well-shaped elements.

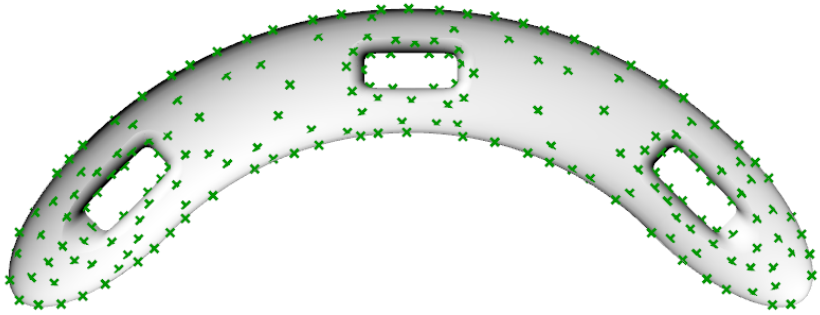


Fig.7 Randomly generated points on a surface for a facade design

Physics-based methods have been effectively used to adjust the distribution of points to achieve more uniform patterns. A bubble-packing method is adopted here to uniformly distribute the randomly generated points on the surface directly, rather than using grid mapping from the plane to the 3D space.

When initial points are generated in the design domain, the bubble-packing method defines proximity-based, repulsive/attractive inter-nodal forces between each pair of points. A dynamic simulation is then conducted to reach a force-balanced configuration for the points. The algorithm is developed in accordance with the observation that a pattern of tightly packed spheres on a surface mimics Voronoi polygons. With Voronoi polygons in the design domain as a starting point, well-shaped Delaunay triangles can easily be created by connecting the centers of the tightly packed spheres, constructing the dual. This paper adopts such a bubble-packing method to uniformly distribute the randomly generated points before connecting them to produce a grid.

4.1 Bubble-packing method

The main assumption of a bubble-packing method in this context is that the nodes in the grid are assumed to be elastic bubbles. In a physically-based dynamic simulation, the overlaps between bubbles generate inter-bubble forces $\vec{T}_{ij}$, defined as:

$$\vec{T}_{ij} = \begin{cases} k_b \cdot (\frac{D_i + D_j}{2} - |\vec{d}_{ij}|) \cdot \frac{\vec{d}_{ij}}{|\vec{d}_{ij}|}, & |\vec{d}_{ij}| < \frac{D_i + D_j}{2} \\ 0, & |\vec{d}_{ij}| \geq \frac{D_i + D_j}{2} \end{cases} \quad (2)$$

In this equation, k_b is the linear constant of bubble springs; D_i and D_j are the diameters of the i-th and j-th bubble respectively, and $\vec{d}_{ij}$ is the vector of displacement from bubble i center to the center of bubble j. To obtain uniform grid edge length, the initial diameters of bubbles are assumed to be a single value D_0, which is one of the important physical assumptions that determines the behavior of the bubble system. To distribute bubbles to the maximum extent over the surface, D_0 should be normally bigger than the desired edge length l_0, such that the balanced bubbles will stay in a static state of being squeezed and the possibility of creating ill-shaped grid elements is significantly lowered. The interactive bubble forces in the i-th bubble are defined as the sum of all the interactive forces from all of the other bubbles in the system (see Eq.(3)), such that a repulsive force is applied when two bubbles are located closer than the stable distance D_0 (bubbles are overlapping). Most of the internal forces are zero since most bubbles are further away from the range of interaction:

$$\vec{T}_i = \sum_{j=1,\ j\neq i}^{n} \vec{T}_{ij} \quad (3)$$

where n *is* the number of simulated bubbles. A force is also used to constrain the bubbles to the surface, so that the centers of the bubbles stay on the surface, as shown in the following equation:

$$\vec{P}_{si} = k_c \cdot \vec{d}_{si} \quad (4)$$

where k_c is a linear spring constant for attracting the bubble to the surface and is much larger than the force between bubbles, $\vec{d}_{si}$ is the displacement vector from p_i, the location of the center of bubble i, to its closest point on the surface. The attracting forces are applied to make sure that all the centers of bubbles stay on the given surface.

A resistance force, f_i, of intermediate magnitude, is also imposed on the bubbles to simulate viscous damping, and depends on the bubble velocity and acts in the opposite direction to that of the bubble's motion:

$$\vec{f}_i = -k_f \cdot \vec{v}_i \quad (5)$$

where k_f is the damping coefficient; $\vec{v}_i$ is the velocity vector of the i-th bubble.

As mentioned above, all the forces acting on a bubble come from the interactions with other bubbles, attraction to the surface and viscous damping. The resultant force $\vec{F}_i$ acting on the i-th bubble is therefore calculated as the sum of these components:

$$\vec{F}_i = \vec{T}_i + \vec{P}_{si} + \vec{f}_i \tag{6}$$

where $\vec{F}_i$ is acted though p_i, the center of the i-th bubble.

4.2 Dynamic relaxation

Given the inter-bubble forces defined above, the objective is to find a bubble configuration that produces a static force equilibrium. Based on the force equations and Newton's equations of motion, a dynamic simulation procedure that assumes a lumped mass at the center of each bubble is employed to solve the force balancing problem. The integration of these equations proceeds as follows:

(1) Define n bubbles in the design domain, with their center coordinates p_i (x, y, z), and a lumped mass m for each bubble.

(2) At time $t = 0$ the bubbles are static with an initial zero velocity (i.e. $\vec{v}_i(t=0) = 0$).

(3) At time $t > 0$ the unbalanced and residual force $\vec{F}_i(t)$ produces an acceleration $\vec{a}_i(t)$ for the i-th bubble:

$$\vec{a}_i(t) = \frac{\vec{F}_i(t)}{m} \tag{7}$$

(4) From the current positions $\vec{s}_i(t)$, velocities $\vec{v}_i(t)$ and accelerations $\vec{a}_i(t)$ of each bubble, the new positions and velocities at time ($t+\delta t$) can be predicted:

$$\vec{v}_i(t+\delta t) = \vec{v}_i(t) + \vec{a}_i(t)\delta t \tag{8}$$

$$\vec{s}_i(t+\delta t) = \vec{s}_i(t) + \delta\vec{s}_i(t) = \vec{s}_i(t) + \frac{\vec{v}_i(t) + \vec{v}_i(t+\delta t)}{2}\delta t \tag{9}$$

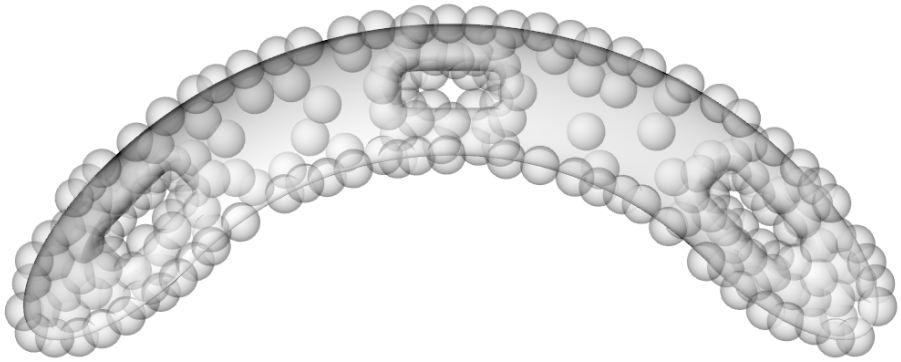
where δt is the (small) time step.

(5) The process is repeated from step (3), using the positions and velocities at time $t+\delta t$ to calculate forces $\vec{F}_i(t+\delta t)$ and accelerations $\vec{a}_i(t+\delta t)$ at time $t + \delta t$. The cycle is repeated until $|\delta\vec{s}_i(t)|$ is acceptably small.

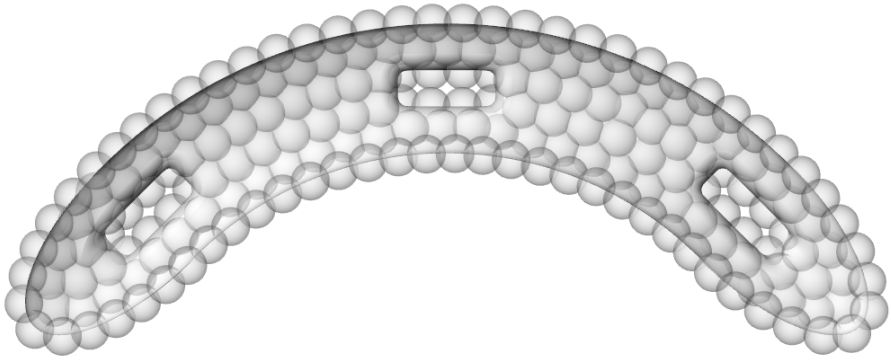
With the above equations of motion, the remaining task is to determine a combination of physical parameters, namely, the mass m, the damping coefficient k_f, and the linear spring constant k_b of inter-bubble force. Though this requirement is not often discussed in physically based approaches, it clearly constitutes an important issue, since the characteristics of the system are all determined by the above parameters. If these parameters are not chosen carefully, the system may be very slow to converge, oscillatory, or in the worst case, completely unstable. As a result, it is vital to select suitable values for these parameters so that the system strikes a balance between stability and efficiency. It is recommended by Shimada and Gossard [17] that the physical parameters of the bubble system should be chosen to satisfy the following relationship:

$$\xi = \frac{k_{\mathrm{f}}}{2\sqrt{m \cdot k_{\mathrm{b}}}} \approx 0.7 \tag{10}$$

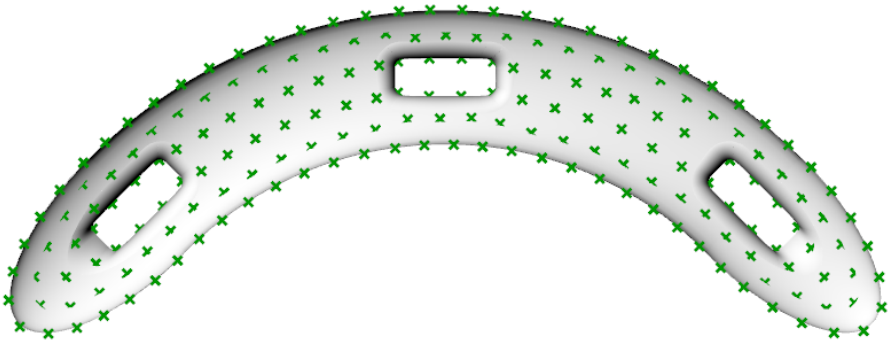
The bubbles reach an equilibrium that is closely packed, within a relatively short number of steps, due to the action of resistance forces due to viscous damping. Fig.8(a) illustrates the initial bubble distribution, while Fig.8(b) shows the resulting bubbles in equilibrium over the surface. When the center-points of the bubbles are extracted from Fig.8(b), the results are as presented in Fig.8(c), which shows clear improvement over that the initially generated points (Fig.8(a)) in terms of uniformity of distribution.



(a) Initial bubbles



(b) Bubbles in equilibrium



(c) Uniformly distributed nodes

Fig.8 Discretization of the facade surface using the bubble-packing method: (a) Initial nodes and bubbles; (b)

Bubbles in equilibrium and (c) Uniformly distributed nodes

4.3 Triangulation

The generated points are connected into a triangular grid according to a Delaunay Algorithm [23]. The procedure is concluded as follows:

(1) Calculate the Voronoi diagram of the bubble centers obtained in Section 3.2. A Voronoi diagram is a partition of space into domains based on the distance to given vertexes in a specific subset of the space. For each vertex, there is a corresponding domain consisting of all points closer to that vertex than to any other.

(2) Compute the intersection curves between the Voronoi diagram and the surface *S*, which forms a Voronoi-like diagram on the surface (Fig.9). More efficient algorithms can be found in [23-25].

(3) For each edge of the Voronoi diagram, connect the two adjacent points closest to the corresponding edge. Traverse all edges of the Voronoi diagram and get the dual graph of the Voronoi diagram, namely the Delaunay triangulation on the surface.

(4) Finally, delete the faces of the Delaunay triangulation that are outside the surface. The grid based on the balanced bubble centers is thus obtained (Fig.10).

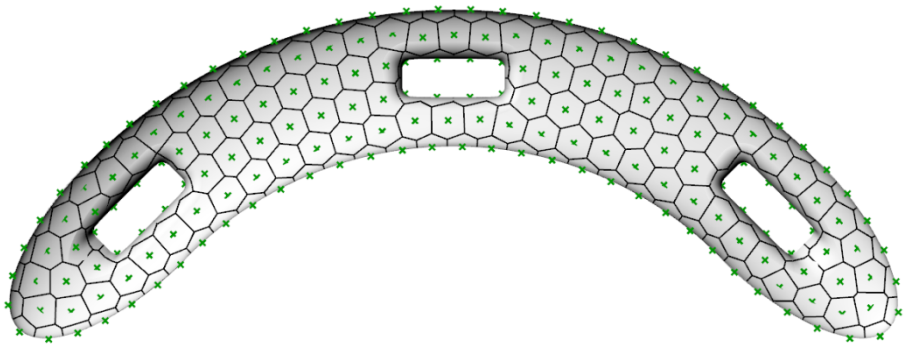


Fig. 9 Voronoi diagram on the surface with 191 vertices

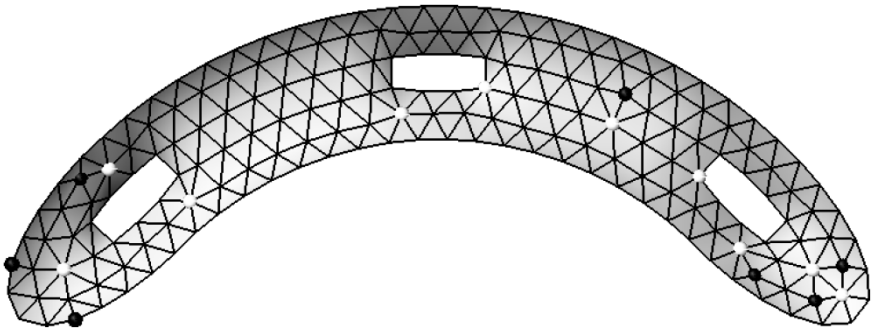


Fig.10 Triangular grid based on optimized node placement (black nodes are connected with 5 edges and white nodes are connected with 7 edges)

It is shown in Fig.10 that the generated triangular grids are generally uniform and well-shaped, with most triangles close to being equilateral. However, there are a number of interior nodes that are connected by 5 or 7 edges (the black and white nodes in Fig.10), which causes the grid to appear to lack fluency, and a method is therefore required to obtain more nodes with 6 connected edges, which would also ideally ensure more edges met at a node with an angle of 60°, decreasing the number of more acute or obtuse angles. In this case, the generated grid will become more regular and it seems likely that triangle shapes would be further improved. Section 4 will use edge operations on the grid to improve its regularity and Section 5 will introduce indexes to measure its quality.

5 Connectivity regularization

With the initial point placements optimized and a uniformly-sized and well-shaped grid obtained, the next step of the grid generation framework is to propose an effective algorithm to improve the grid regularity by adjusting its connectivity. Topologically, a closed mesh with nonzero Euler characteristic must exist at least one irregular vertex. The irregular vertices naturally appear in high discrete Gauss curvature regions in the mesh. Li et al. [26] proposed an interactive framework to control the type, location, and the number of irregular vertices in a triangle mesh. An improved algorithm based on Li et al. [26] is proposed to modify the grid connectivity, specifically for architectural design while considering the impact of boundary curves.

The algorithm employs a series of edge operations such as edge-flip, edge-collapse and edge-split [27]. For each edge, a regularization index is computed that depends on the degrees of its end points and opposite points. Any edge for which this index exceeds a prescribed threshold will be adjusted until a more regular grid is obtained.

5.1 Optimization objective

The degree, d_i, of the i-th vertex in a triangulation, is the number of edges connected to it. d_{oi} is the optimal degree of node i. For an equilateral triangular grid, the interior nodes have optimal degree $d_{oi} = 6$, and for boundary vertices, d_{oi} is related to the boundary curvature and usually equals 4 to obtain 60° angles. To simplify the computational process, adjustments are made to boundary vertices by assigning virtual degrees [27]. As shown in Eq.(11), the adjusted degree at a boundary vertex is equal to the actual degree plus the virtual degree, which allows the target adjusted degree, d_i^*, to be the same for all points in the system:

$$d_i^* = \begin{cases} d_i, & p_i \in P_{\text{int}} \\ d_i + v_i, & p_i \in P_{\text{bou}} \end{cases} \tag{11}$$

where d_i^* is the adjusted degree for the i-th vertex, P_{int} and P_{bou} are the interior and boundary vertices respectively

and v_i is the virtual degree, defined as the following:

$$v_i = \begin{cases} 4-k, & \alpha_k < \theta_i \le \alpha_{k+1}, k = 0,1,2,3,4 \\ 0, & \alpha_5 < \theta_i \end{cases} \tag{12}$$

where $a_k = 60°\sqrt{k(k+1)}$, $k = 0,1,2,3,4,5$ and θi is the sum of the interior angles (in degrees) of the faces at the i-th vertex.

For example, the simple grid in Fig.11 has a boundary point labelled p2 which has nodal degree d_2 = 5 and is surrounded by 4 faces (f1-f4). The interior angles at p2 sum to θ_2 = 57° + 66° + 64° + 34° = 221°. As $\alpha_3 = 208° \le \theta_2 = 221° \le \alpha_4 = 268°$, therefore k = 3 and the virtual degree of p2, $v_2 = 4 - 3 = 1$. As a result, the adjusted degree of the boundary point is $d_2^* = 6$. The degrees of other vertices in Fig.11 are also shown in Table.1.

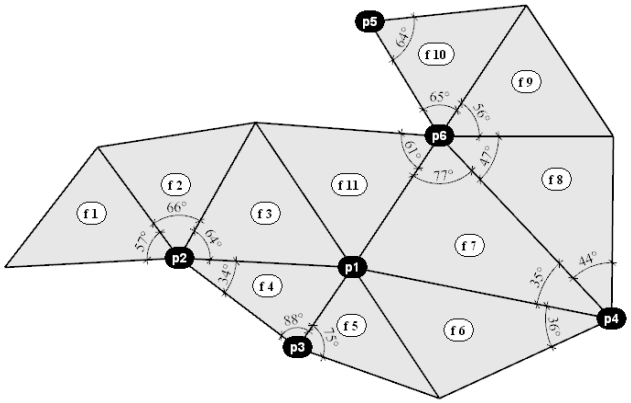


Fig.11 A simple example to illustrate the calculation of degree at each vertex

Table.1 Degrees for vertices in the grid shown in Fig.11

Vertex number i	1	2	3	4	5	6
Actual degree d_i	6	5	3	4	2	6
Virtual degree v_i	\	1	2	3	4	0
Adjusted degree d_i^*	6	6	5	7	6	6

A vertex p_i is regular if $d_i^* = 6$ and irregular if $d_i^* \neq 6$. For better comparison, vertices in the following figures are colored according to their adjusted degree, a vertex is labeled as black if $d_i^* < 6$ and white if $d_i^* > 6$. Such vertices are non-ideal, and will decrease the regularity and fluency of the grid and produce wrinkles. Therefore, improving the regularity of a grid means minimizing the nodal degree residual (i.e. the divergence from all nodes having adjusted degree 6), defined as the function $R(G)$:

$$\gamma_i = \begin{cases} \lambda, & p_i \in P_{\text{int}} \\ 1-\lambda, & p_i \in P_{\text{nak}} \end{cases} \tag{13}$$

$$R(G) = \sum_{i=1}^{n} \left(d_i^* - 6\right)^2 \tag{14}$$

where n is the total number of grid vertices; γ_i is the weight of i-th vertex; λ is the weight of interior vertices, and $\lambda \in [0,1]$. The residual will reflect how much the mesh differs from the ideal equilateral triangular grid. However, the irregular boundary vertices have less impact on the overall fluency of the grid, therefore, λ tends to be larger than 0.5. Studies carried out during the preparation of this paper has suggested that $\lambda = 0.9$ leads to good quality grids, and as such this value has been assumed for the rest of the work reported here.

5.2 Edge operations

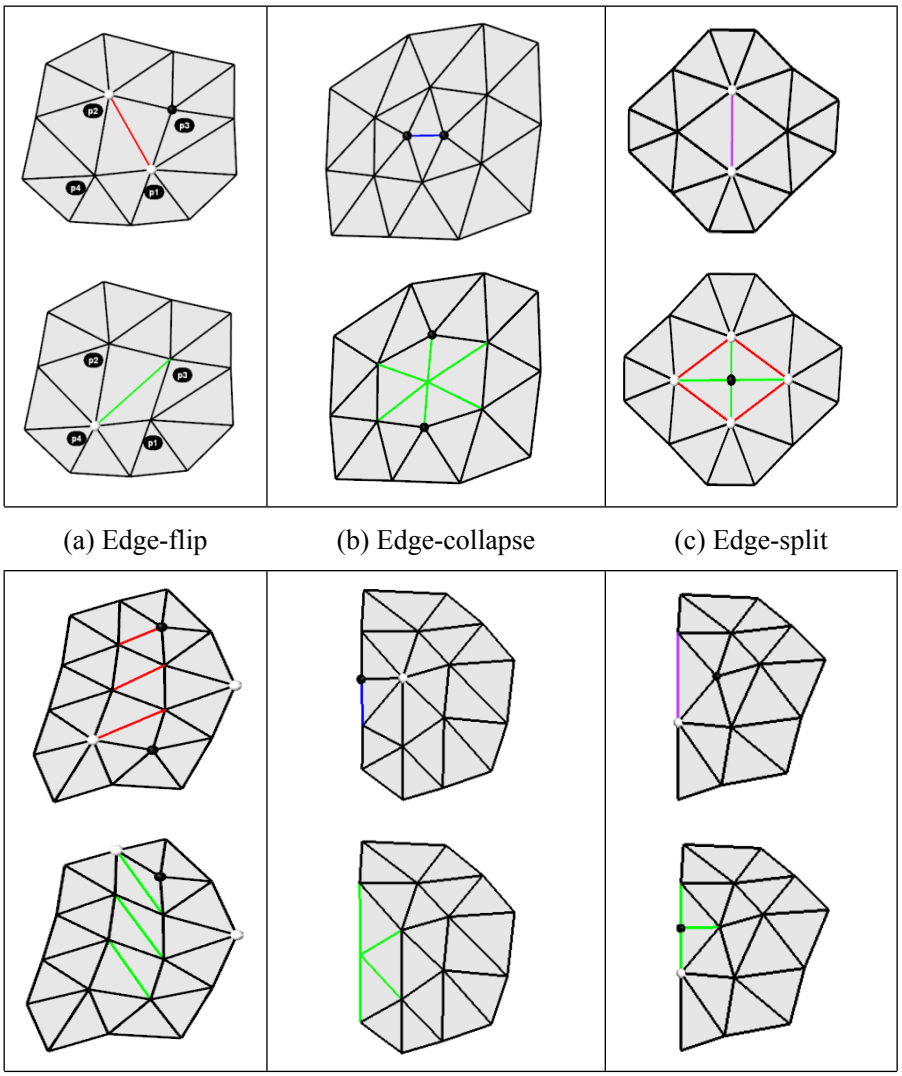
The regularity of an edge depends on the number of non-ideal vertices at its ends. Potential edge operations are carried out if any one of its end vertices is non-ideal. The regularization index of an edge depends on the adjusted degrees of its end vertices and its two opposite vertices. Taking Fig12(a) as an example, the regularization index of the red edge depends on the degrees of vertices p1-p4 and the nodal degree residual of an edge before and after the edge operation are defined in Eq. (15) and Eq. (16), respectively. The change in nodal degree residual of an edge before and after an edge operation ΔR_j is defined in Eq.(17). The edges with sma aller value of ΔR_j will be operated in priority. As shown in the lower part of Fig.12 (a), when an edge is flipped, the degree of the four vertices (p1-p4) is changed. ΔR_j is calculated from Eq.17 and if $\Delta R_j < 0$, the edge-flip has been effective in reducing $R(G)$.

$$R_j = \sum_{i=1}^{4} \gamma_i (d_i^* - 6)^2 \tag{15}$$

$$R_j' = \gamma_1 (d_1^* - 7)^2 + \gamma_2 (d_2^* - 5)^2 + \gamma_3 (d_3^* - 7)^2 + \gamma_4 (d_4^* - 5)^2 \tag{16}$$

$$\Delta R_j = R(G') - R(G) = R_j' - R_j \tag{17}$$

where d_i^* is the adjusted nodal degree of p_i in the original grid and i =1,2,3,4; G and G' are the grid before and after the edge-operation respectively. For an edge with non-ideal endpoints, the edge might benefit from operations including edge-flip, edge-collapse and edge-split, as shown in Fig.12. The edge-flip procedure, shown in Fig.12(a), can be beneficial when the two endpoints of the edge are white ($d_i^* > 6$) and one of its adjacent point is black ($d_i^* < 6$). When both endpoints are black, the edge needs be collapsed to increase the nodal degree at both vertices, as shown in Fig.12 (b). If both its vertices are white but the two adjacent vertices are ideal, this edge should be split as shown in Fig.12 (c). After splitting, two of its neighboring edges will need to be flipped according to the rule defined in Fig.12(a). It can be seen that edge-collapse and edge-split operations usually will not generally decrease the nodal degree residual $R(G)$ directly. Nevertheless, the distribution of non-ideal vertices is changed and further edge-flip operations on neighboring vertices then become effective.



(a) Edge-flip (b) Edge-collapse (c) Edge-split

(d) Edge-drift (e) Boundary edge-collapse (f) Boundary edge-split

Fig.12 Types of edge-operations

If the endpoints of an edge both are irregular (colored), the edge is a featured-edge. Specifically, if one endpoint of an edge is a white while the other endpoint is black, the edge is termed a *drifting* edge. The operations allow the movement of the white and black vertices across a regular region of the grid. When drifting edges meet other non-regular vertices or featured edges, effective edge-flips, edge-collapse or edge-split can be conducted and the nodal degree residual decreased. As shown in Fig.12 (d), when a featured edge with one white vertex and one black vertex drifts to the boundary, with one boundary vertex, operations can be carried out to reduce $R(G)$ further. If the boundary endpoint is black, the singularity of both endpoints can be eliminated by collapsing a boundary edge (Fig.12 (e)). If the boundary endpoint is white, the singularity of the interior endpoint can be transferred to a boundary vertex by splitting a boundary edge (Fig.12 (f)). As λ<1, both the boundary edge-collapse and edge-split operations can

reduce the nodal degree $R(G)$. Fig.13 illustrates this in practice, where the featured edges and vertices of the large-span space structure surface can be identified. However, the edge-collapse and edge-split operations will change the number of vertices n. As n is related to the grid size, n is usually limited to a certain range, which is a constraint to the optimization. The edge-collapse and edge-split operations can therefore only be performed whilst n does not exceed the range.

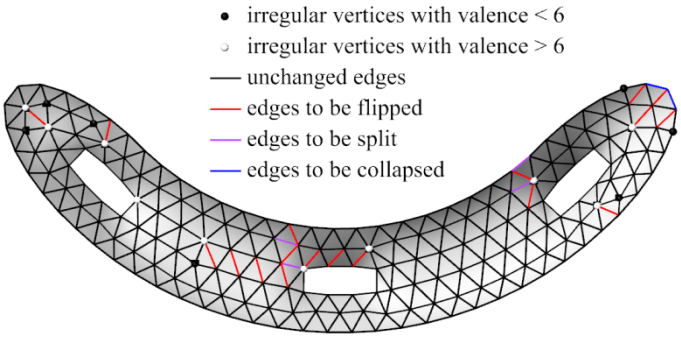


Fig. 13 Edges to operate

The algorithm to regularize connectivity requires a series of local operations, including edge-flips, edge-collapses and edge-splits and edge-drift. In the authors' software implementation, the irregular edges are stored in an ordered queue, and the edges with the smallest ΔR_j are given the highest priory. After each edge-modification the queue priorities are updated. That is, the algorithm always performs effective edge-flips on irregular edges according to their priority. Once edge-flips have been applied, edge-drift, edge-collapse and edge-split operations are performed. These steps are repeated until there are no featured edges except drifting boundary edges on the boundary.

Grid optimization with the objective to minimize R(G) as described above can easily converge to a local minimum rather than global minima. At this point, there may be some isolated irregular vertices among the generated grid. Based on the simulated degradation method, one technique is to perform edge-collapse or edge-split to the edges connected to the isolated irregular vertices randomly, which will cause R(G) to rise, and then perform conventional optimization to reduce R(G). The irregular vertices can, therefore, be transmitted to the boundary of the grid or even be eliminated directly. However, the computational cost of this algorithm for the search of global minima is high [28] and the edge segmentation or merging will change the number and distribution of nodes, which will in turn decrease the uniformity of the gird. An alternative method is to allow users to specify the operations for isolated irregular vertices and to further optimize the grid in an interactive way within the framework. This method requires only a limited amount of manual intervention and can rapidly regularize the grid. The method is outlined by Algorithm 1and a work-flow is presented in Fig 14.

1

Algorithm 1 connectivity regularization:

Input: G, an irregular grid
Output: G, the adjusted grid

while the maximum number of loops is not reached:
    dR_list ← enumerate all edges to calculate ΔR
    sort(dR_list), from small to big
    while dR_list [0]<0:
        ce ← dR_list [0]'s corresponding edge
        flip(ce)
        update dR_list
    end while
    R_list ← enumerate all edges to calculate R
    sort(R_list), from big to small and if equal, corresponding edge with two irregular vertices in same type ahead
    if R_list [0]>=2:
        ce ← R_list [0]'s corresponding edge
        switch type(ce):
            case two black vertices:
                collapse(ce)
            case two white vertices:
                split(ce)
            case one black one white:
                drift(ce),until encounter other irregular vertices or reach the boundary
        end switch
    else:
        mark all irregular vertices
        the user perfom operations to the edges with Rj >0 interactively
    end if
end while

2

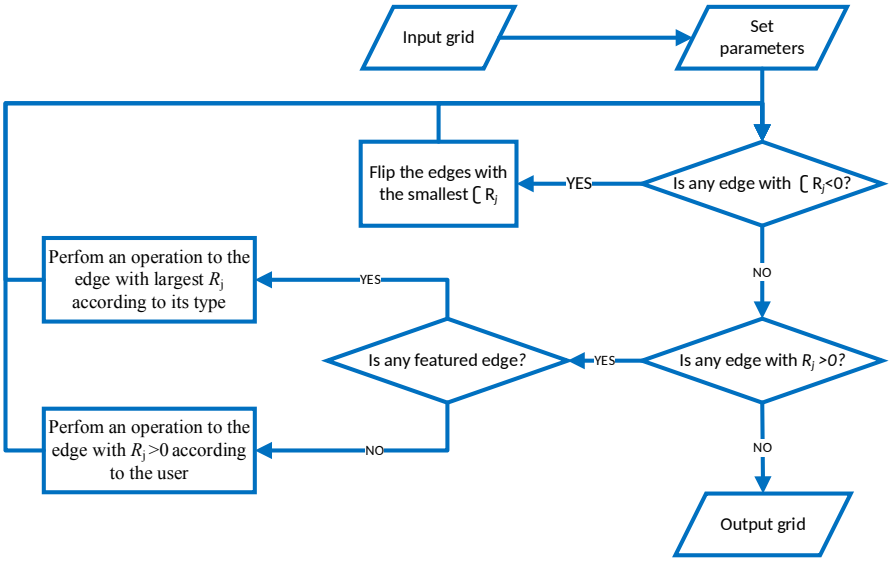


Fig.14 Flow diagram of the algorithm to regularize connectivity

The algorithm to regularize connectivity is incorporated into the authors' framework. When this algorithm is applied to the surface in Fig. 13, a grid with fewer irregular vertices is found, as shown in Fig.15, which in this case has managed to remove all interior vertices with valence ≠ 6, leaving irregular vertices only on the boundary.

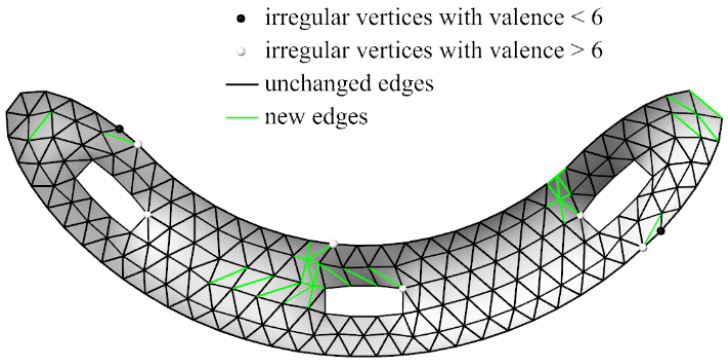


Fig.15 Regularized grid

6 Grid relaxations

A consequence of the edge operations above is that, whilst improving the topological regularity, the triangles of the resulting grids become less equilateral. A net-like method is therefore used to re-smooth the generated grid on the surface without changing its improved topology. The net-like method, which is also physically based, is similar to the principle of the bubble-packing method. However, the net-like method only takes into account the interaction between connected nodes, and only adjusts the placement of these nodes by a relatively small amount. The objective

of the net-like method is to relax the grid to achieve the overall uniformity of grid size without changing the topology of the grid.

The triangular grid is treated as a net of springs. Each edge is assumed to be a linear spring, and each vertex is modelled as a particle with a lumped mass m. For the i-th particle p_i, there are n_i particles connecting it, and the resultant spring force at particle p_i is calculated by Eq.(18) and Eq. (19):

$$\vec{T}_{ij}' = k_b(l_0 - |\vec{l}_{ij}|) \cdot \frac{\vec{l}_{ij}}{|\vec{l}_{ij}|} \tag{18}$$

$$\vec{T}_i' = \sum_{j=1}^{n_i} \vec{T}_{ij}' \tag{19}$$

where k_b is the linear spring constant, $\vec{l}_{ij}$ is the displacement vector from the i-th particle p_i to the j-th particle p_j, l_0 is the constant slack-length of the spring and is defined as the average value of all edge lengths across the surface:

$$l_0 = \bar{l} = \frac{\sum_{j=1}^{n_e} l_j}{n_e} \tag{20}$$

where n_e is the total number of edges, and l_j is the j-th edge length.

For boundary vertices, there are also boundary attractive forces defined to keep the boundary particles on the boundary:

$$\vec{P}_{ci} = \begin{cases} k_c \cdot \vec{d}_{ci}, & p_i \in P_{nak} \\ 0, & p_i \in P_{int} \end{cases} \tag{21}$$

where k_c is a linear spring constant for the attractive force of the boundary curve, and k_c is much larger than k_b; $\vec{d}_{ci}$ is the displacement vector from p_i to its closest point on the curve.

Similar to the bubble-packing model presented in Section 3.1, the surface attractive force, the boundary anchoring force and the viscous damping forces are summed to compute the resultant force acting on the i-th particle:

$$\vec{F}_i = \vec{T}_i' + \vec{P}_{ci} + \vec{P}_{si} + \vec{f}_i \tag{22}$$

The dynamic simulation procedure presented in Section 3.2 is once again employed to solve the dynamic motion equations presented in Eqns.(7)-(9) to obtain a static force equilibrium of the net-like system. For the triangular grid in Fig.15, the method is employed to relax the grid and the result is shown in Fig.16. It can be seen that the new grid is more uniform and fluent, and looks very suitable for free-form grid structure design, but a more robust measure of this suitability is required to allow formal comparison.

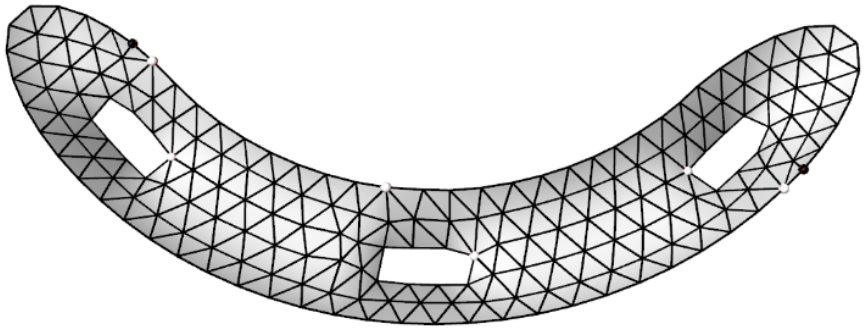


Fig.16 Relaxed grid

7 Grid quality evaluation

Architects or engineers usually evaluate grids based on visual checks. This requires the designers' experience to assess the quality of grids in terms of uniformity, regularization and fluency. Quantitative methods are therefore essential to evaluate the quality of architectural grids of free-form surfaces. Grid quality indices, such as face shape quality and edge length, can give an overall description of the quality, but these indices are once again mainly focused on FEA applications and have not been derived in the context of structural gridshells. The authors therefore present a new index to assess the fluency of triangular grids, whereby improved fluency means a better visual expression of a grid structure as required in most architectural applications.

To quantify the grid quality in terms of edges, the coefficient of variation is used based on statistical principles [29]:

$$s^2(l) = \frac{\sum_{i=1}^{n_e} (l_i - \bar{l})^2}{n_e - 1} \tag{23}$$

$$r = s(l)/\bar{l} \tag{24}$$

The smaller the coefficient r, the more uniform the edge lengths.

To evaluate the grid quality in terms of triangles, the triangular shape index q, defined by Eq.(25), is used. Since equilateral triangles are desired in a grid structure design, a larger mean value of the triangular shape index implies the better shape of the triangular grid:

$$q = 4\sqrt{3}\frac{A_t}{l_a^2 + l_b^2 + l_c^2} \tag{25}$$

where A_t is the triangle area; l_a, l_b and l_c are the side lengths of the triangle. $q \in [0,1]$, and for an equilateral triangle $q = 1$, while for a degenerate triangle (three points collinear), q is taken as zero.

Grid fluency is an important aspect in evaluating a free-form gridshell, however there is no previously proposed standard and qualitative measurement for the assessment of grid fluency. An index is therefore proposed to evaluate the fluency of a triangular grid. As discussed in Section 4, the nodal degree residual $R(G)$ of a gridshell is related to

the fluency of a grid, the average nodal degree residual, μ, is therefore calculated from Eq.(26) and (27):

$$n^* = \lambda n_{\text{int}} + (1-\lambda) n_{\text{bou}} \tag{26}$$

$$\mu = \sqrt{\frac{R(G)}{n^*}} \tag{27}$$

where $R(G)$ is defined in Eq.(14), n_{int} is the number of interior vertices , n_{bou} is the number of exterior vertices and $n_{\text{int}}+n_{\text{bou}} = n$. It is worth noting that n^* is the number of grid vertices adjusted by λ to differently weight the boundary and interior node degree.

For a node with degree 6 (an ideal point), another factor that can affect the fluency is the angle between two opposite edges and the opposite angles at the point. As shown in Fig.17, if the edges E1 and E4 are in a straight line (i.e $\beta_{1,4} = 180°$), a more fluent triangular grid will be achieved. It is also expected that the opposite angles between edges E1 and E2 (β_1) and the angle between edges E4 and E5 would be the same in a more fluent grid.

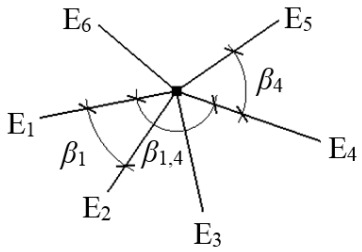


Fig.17 Angles of a node

As a result, an angle fluency index δ of an interior regular vertex is defined by Eqns.(28)-(30).

$$\sigma_i = \sqrt{\frac{\sum_{j=1}^{3} (\beta_{j,k} - 180°)^2}{3}}, k = j+3 \tag{28}$$

$$\tau_i = \sqrt{\frac{\sum_{j=1}^{3} (\beta_j - \beta_{j+3})^2}{3}} \tag{29}$$

$$\delta_i = \sqrt{\sigma_i^2 + \tau_i^2} \tag{30}$$

where δ_i denotes the angle fluency index of the *i*-th vertex; $\beta_{j,k}$ denotes the angle between the *j*-th edge and the *k*-th edge and β_j denotes the angle between the *j*-th edge and the (*j*+1)-th edge, as shown in Fig.17. The smaller the angle fluency index of a regular vertex, the more fluent the grid around that vertex.

A simple grid in Fig.18 is evaluated by using this index as an example. p2 is an ideal point with 6 equilateral triangles, the angle fluency index is $\delta = 0°$. The points p3 and p9 are visually non-ideal, and the angle fluency index is as large as 40.7° and 53.9°, respectively. As shown in Table.2, the magnitude of δ_i reflects the grid fluency around each

point *i*. Fig.19(a) and Fig.19(b) mark the vertices of the grid before and after the grid relaxation process described in section 5, coloring the vertices according to their angle fluency indexes. It can be seen that the grid relaxation process significantly smooths the grid as the angle fluency indexes of the vertices reduce obviously.

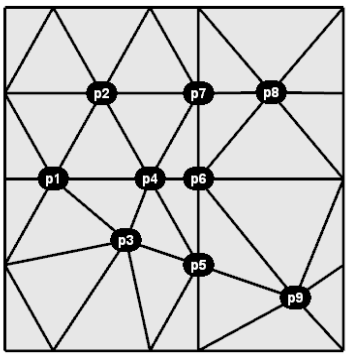
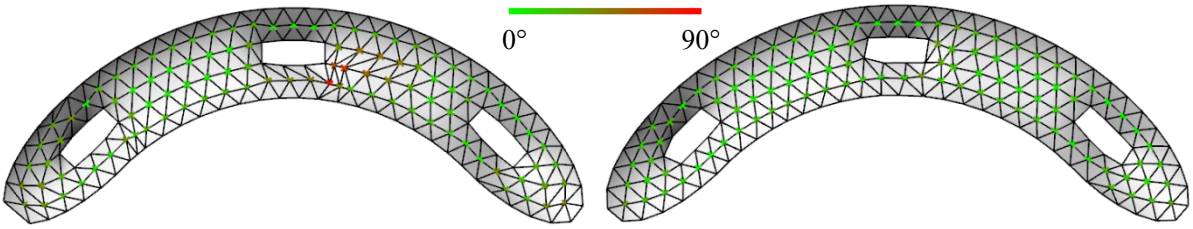


Fig.18 A planar grid with 9 nodes

Table.2 Fluency index of interior vertices

Node number	p1	p2	p3	p4	p5	p6	p7	p8	p9
σ (°)	16.8	0.00	28.8	6.90	24.5	32.8	24.5	0.841	38.3
τ (°)	11.9	0.00	28.7	4.90	24.5	32.8	24.5	0.800	37.9
δ (°)	20.6	0.00	40.7	8.46	34.6	46.4	34.7	1.18	53.9



(a) Regularized grid (b) Relaxed grid

Fig.19 Angle fluency indexes of interior vertices

The overall fluency index, Q, of a grid is defined in Eq.(31) by including the nodal degree residual and angle fluency index taken from Eq.27 and Eq.30 respectively. The fluency index Q is therefore used to make a comprehensive evaluation of the quality of the entire grid, with the larger the value of Q, the more fluent the generated grid structure.

$$Q = \frac{1}{\mu + k_Q \rho \bar{\delta}} \tag{31}$$

where the coefficient $k_Q = \frac{1}{60^\circ}$ to make the units comparable and the proportion of the interior regular vertices, ρ, is defined as:

$$\rho = \frac{\lambda n_{\text{inr}}}{n^*} \tag{32}$$

where n_{inr} is the number of interior regular vertices.

8 Case studies

Case studies are presented in this section to illustrate the application of the proposed framework. For case studies, the linear spring constant of bubbles k_b is set to be 10. The original diameters of bubbles d_0 are assumed to be 1.2 times of the desired length l_0. The linear spring constant k_c for attracting the bubble to the surface is set to be 100 while the damping coefficient k_f is set to be 1.

8.1 Hexagon triangulation

Fig.20 presents a simple benchmark example of a hexagonal design domain in which 61 points are randomly generated. The bubble-packing method was firstly employed to regularize the nodal placement, as shown in Fig.21 (a). The Delaunay-like triangulation was then used to generate the initial triangles in Fig.21(b). Connectivity optimizations were subsequently conducted in Fig.21(c) to eliminate the non-ideal internal points. However, at this stage the grid edges are no longer uniform, and the net-like method was employed to relax the grid and the result is shown in Fig.21(d). It can be seen from Fig.21 that the expected regular triangulation is achieved using the framework presented above.

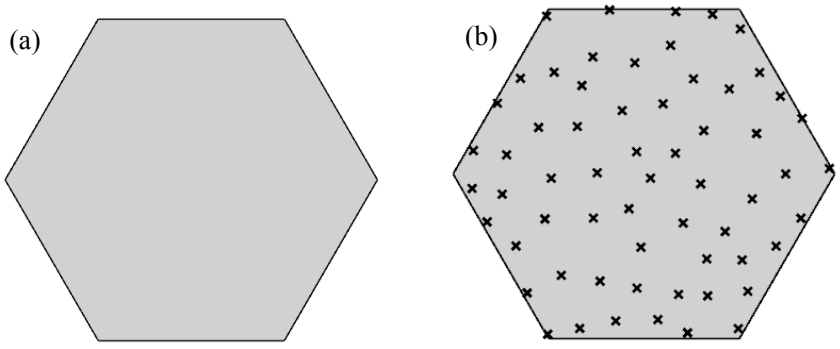


Fig.20 Given surface for grid generation: (a) A hexagon design domain; and (b) 61 randomly generated points

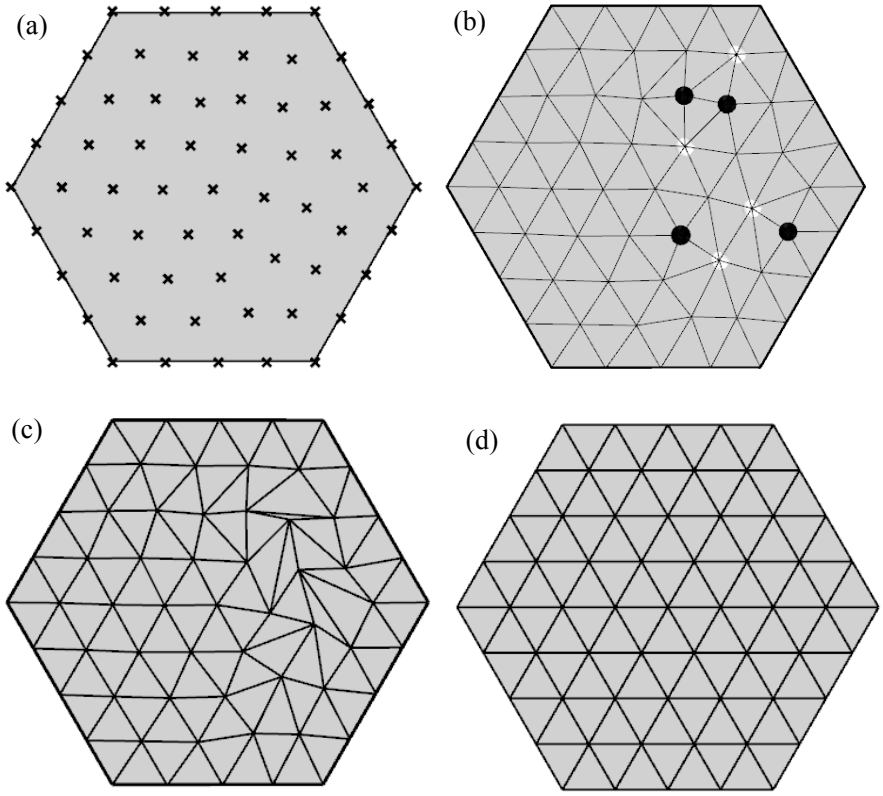


Fig.21 Grid generation procedures for the triangulation of a hexagon design domain with 37 internal points and 24 boundary points: (a) Point uniformization; (b) Triangulation; (c) Connectivity optimization and (d) Grid relaxation

Fig.22 investigates the influence of the number of initially generated points. In the grid shown in Fig.22(a), precisely 36 interior points were inserted, and all non-ideal points were eliminated using the connectivity optimization. The grid relaxation has also been employed to improve the smoothness of the grid. In the grid shown in Fig.22(b), 38 internal points were inserted. Neither of these grids stabilized to eliminate all the irregular connections. The black and white nodes presented are located and oriented in such a way that further refinement is impossible to improve the node connectivity, even if continuing to apply the edge operations indefinitely.

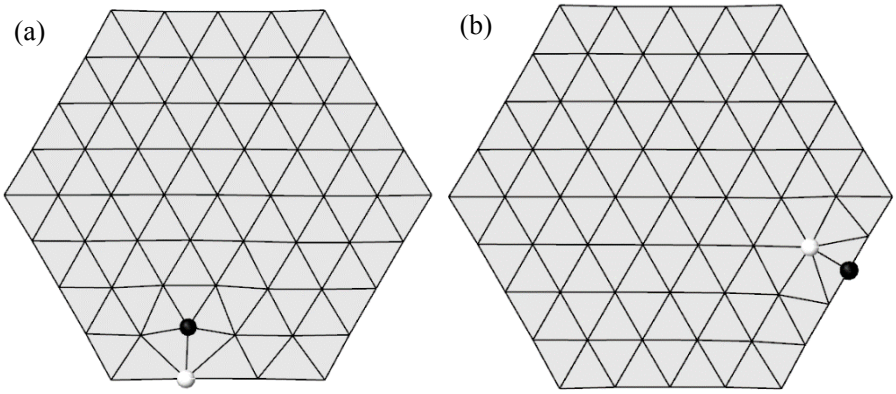


Fig.22 The influence of the number of initial points (a) 60 points and (b) 62 points

8.2 A gridshell roof

Fig.23 presents a more complex example for the grid generation over a free-form surface for a gridshell roof taken from a real project. It is 78 *m* in length, 39 *m* in width and 12 *m* in height and contains 2 closed voids to provide light to the building below and one open void as an entrance. The grid generation framework proposed in this paper has been used to generate the grid for the design. The bubble-packing method was first employed to smooth the randomly generated set of points and Delaunay triangulation was used to connect the points. From the results, shown in Fig.24, it is clear that the degree of a significant number of the connections is non-ideal and the grids are therefore not fluent. In Fig.25, the final grid, optimized using the authors' scheme, is not only very uniform, but also dramatically increases in fluency. The result is likely to be preferred by the architects due to its more aesthetic visual appearance.

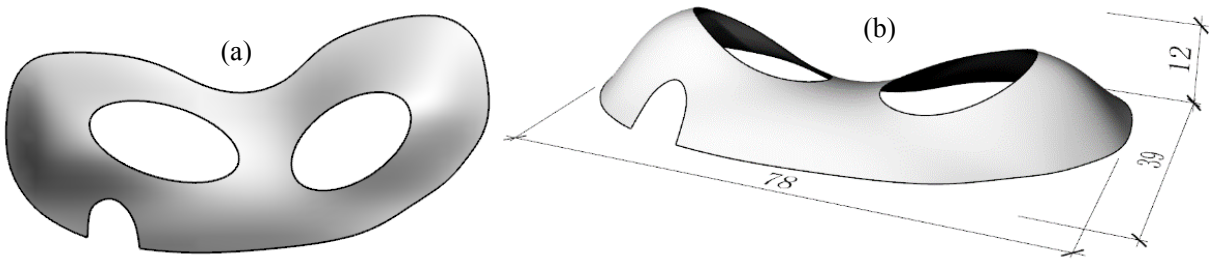
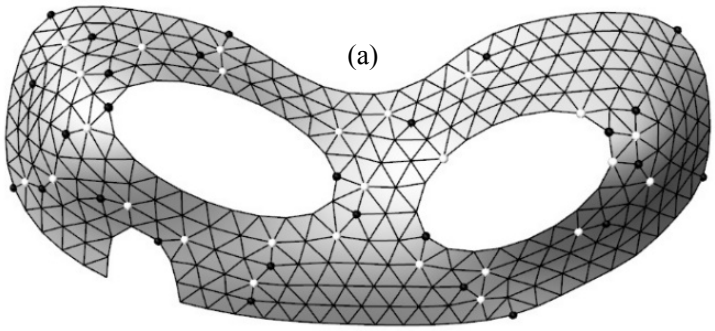


Fig. 23 A free-form surface with complex boundaries: (a) Top view and (b) Perspective view (units: *m*)



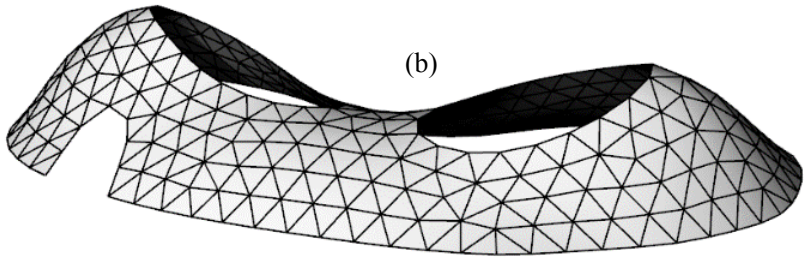


Fig.24 Grid generated by the bubble-packing method: (a) Top view and (b) Perspective view

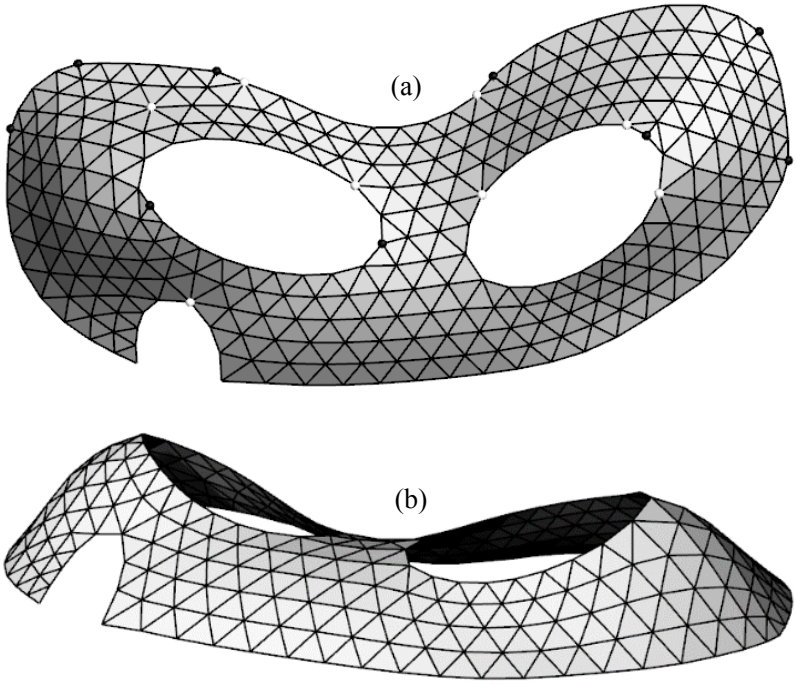


Fig.25 Grid by the connectivity regularization and net-like method: (a) Top view and (b) Perspective view

For comparison, a guide-line method based on surface flattening is employed to generate grids over the surface. The method flattens the surface onto a plane, arranges a uniform grid on the planar surface, and controls the grid generation direction with a guide line. A spatial grid is then obtained by mapping the grid in the plane back onto the surface, as shown in Fig.26. The blue areas are meshes trimmed by the boundary curves. The generated grids possess good fluency and uniformity generally, but the meshes near the boundary are not uniform and the number of nodes is large, which makes it difficult to be directly employed for the design of practical grid structures.

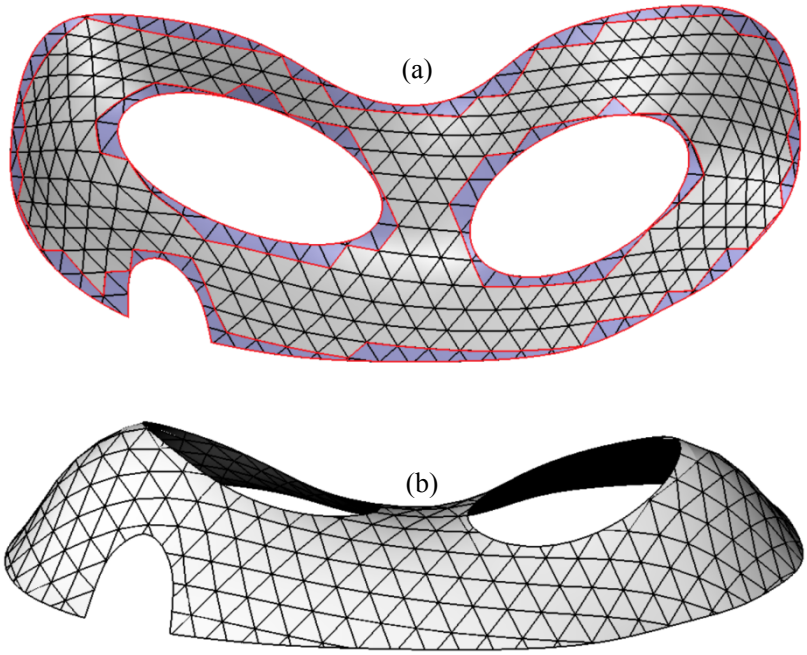


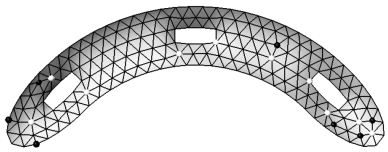
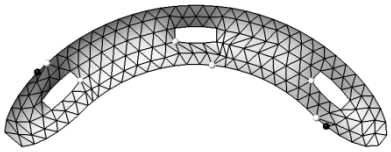
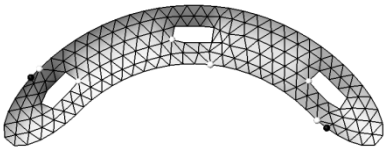
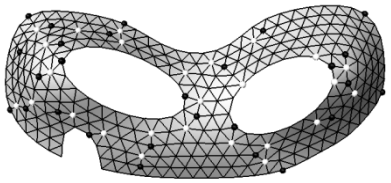
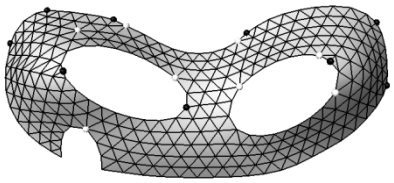
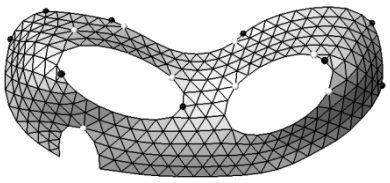
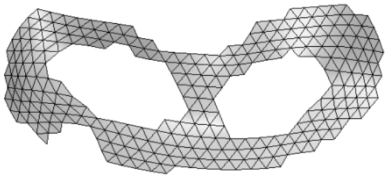
Fig.26 Grid by the guide line method: (a) Top view and (b) Perspective view

The grid quality indices for the facade of the space roof structure (roof 1) and the gridshell roof (roof 2) are presented in Table 3. As shown in Fig.26, the grid generated by the guide line method exhibits significant polarization, that is the internal grid (gray region) is of very high quality with r =4.85×10-2, $\bar{q} = 0.993$ and $Q = 7.81$ while the grid near the boundary (blue region) is of poor quality. For a surface with such complicated boundary curves, the proportion of the meshes trimmed by the boundary line is relatively large, resulting in a grid with lower quality.

The grids optimized by the proposed scheme are excellent in terms of edge uniformity, grid shape quality and fluency index, as shown in Table 3. The resulting average edge length of both roof 1 (from Section 6) and roof 2 (from Section 8.2) are close to the expected member length of 24.4m and 3.1m, respectively. Compared to the grids generated by only the bubble-packing method [30], the results from the connectivity optimization have smaller fluency index in terms of point degree (improved connectivity) in both roof 1 and roof 2. However, for roof 1, the edge length index and angle fluency index deteriorate due to the presence of more ill-shaped triangles resulting from the edge operations. For both roof1 and roof2, the grids are of relatively excellent shape with $\bar{q} > 0.97$ and uniform edge lengths with $r \leqslant 0.1$.

For both roof 1 and 2, after the grid connectivity optimization, the degree fluency index μ reduces by 75% and 74% respectively. By using the proposed framework, the fluency index of the whole grid Q has been improved by 157% and 118%, for roof 1 and roof 2, respectively.

Table. 3 Grid quality indexes

Cases	Procedures	Results	$\overline{l}$ (m)	$s^2(l)$ (m^2)	r	$\overline{q}$	μ	$\overline{\delta}$ (°)	Q
Roof 1	Bubble-packing method		25.1	4.52	847	0.976	0.351	19.9	1.10
	Connectivity optimization		25.2	11.4	1340	0.942	0.089	26.6	2.03
	Net-like method		24.4	5.30	944	0.980	0.089	17.5	2.83
Roof 2	Bubble-packing method		3.18	0.0751	862	0.972	0.439	14.5	1.61
	Connectivity optimization		3.10	0.150	1250	0.978	0.116	12.2	3.27
	Net-like method		3.09	0.101	971	0.985	0.116	10.8	3.52
	Guide line method		3.01	0.146	485	0.993	0	8.37	7.81

9 Conclusion

To automatically generate grids over a complex free-form surface for architectural design, this paper proposes a new grid generation and optimization framework. The framework relies on a physically-based bubble-packing model and

a geometry edge operation to achieve triangular grids with balanced rod-lengths and regular grid connectivity. Firstly, an appropriate number of vertices are estimated based on the desired rod-length and the area of the free-form surface, and points are randomly generated and distributed over the surface. A bubble-packing model is then employed to smooth the initial points and a Delaunay Triangulation method is used to connect the generated vertices to obtain an initial triangular grid with balanced edge-length. Subsequently, a series of edge operations, namely edge-flips, edge-collapses and edge-splits, are carried out to improve the regularity of the grid connectivity and a grid with a reduced number of irregular vertices is obtained. Finally, a net-like method is employed to re-smooth the grid, and a uniform and fluent grid is consequently acquired. The proposed grid generation framework is robust, effective and can be applied to free-form surfaces with complex boundary curves. In addition to conventional quantitative measurements to evaluate the grid quality in terms of uniform edge lengths and grid cell shape quality, a new index is presented to evaluate the fluency of grid, which is an important requirement for architectural design. Examples have been presented that show the effectiveness of the proposed framework for grid generation and quantitative evaluation. By using the proposed framework, the fluency index of the grid was improved by up to 157% for the presented examples. The proposed framework can generate triangular grids for the design of free-form gridshell with complex boundaries and the framework has been shown to be useful in facilitating the conceptual design process of free-form gridshells in engineering practice.

The proposed framework is not always applicable to grid generation over a surface that has a sharp change in curvature and a small spatial distance between two patches. However, the framework can be adapted to generate a coarse grid, of an appropriate scale, over the surface first, and then subdivide and relax the grid to obtain a relatively uniform and fluent semi-regular grid.

Acknowledgments

This research was sponsored by the National Natural Science Foundation of China under Grant 51678521, 51778558 and by the Natural Science Foundation of Zhejiang Province LY15E080017. The project is also supported by the Foundation of Zhejiang Provincial Key Laboratory of Space Structures, Grant 21705. The authors would like to thank them for their financial support.

References

[1] S. Malek, C.J. Williams, Reflections on the Structure, Mathematics and Aesthetics of Shell Structures, Nexus Network Journal, 19 (2017) 555-563.

[2] D. Shilin, Development and expectation of spatial structures in China [J], Journal of Building Structures, 6 (2010) 006.

[3] W. Kang, Z. Chen, H.-F. Lam, C. Zuo, Analysis and design of the general and outmost-ring stiffened suspen-dome structures, Eng Struct, 25 (2003) 1685-1695.

[4] C.-Y. Cui, B.-S. Jiang, A morphogenesis method for shape optimization of framed structures subject to spatial constraints, Eng Struct, 77 (2014) 109-118.
[5] S. Adriaenssens, P. Block, D. Veenendaal, C. Williams, Shell structures for architecture: form finding and optimization, Routledge, 2014.
[6] C. Jiang, C. Tang, H.-P. Seidel, P. Wonka, Design and volume optimization of space structures, ACM Transactions on Graphics (TOG), 36 (2017) 159.
[7] S. Lo, Dynamic grid for mesh generation by the advancing front method, Computers & Structures, 123 (2013) 15-27.
[8] S.A. Hannaby, A Mapping Method for Mesh Generation, Computers & Mathematics with Applications, 16 (1988) 727-735.
[9] J. Muylle, P. Iványi, B. Topping, A new point creation scheme for uniform Delaunay triangulation, Eng Computation, 19 (2002) 707-735.
[10] Y. Liu, H.L. Xing, Z.Q. Guan, An indirect approach for automatic generation of quadrilateral meshes with arbitrary line constraints, International Journal for Numerical Methods in Engineering, 87 (2011) 906-922.
[11] S.J. Owen, A survey of unstructured mesh generation technology, in: IMR, 1998, pp. 239-267.
[12] P. Alliez, G. Ucelli, C. Gotsman, M. Attene, Recent advances in remeshing of surfaces, in: Shape analysis and structuring, Springer, 2008, pp. 53-82.
[13] A. Kammoun, F. Payan, M. Antonini, Adaptive semi-regular remeshing: A voronoi-based approach, in: Multimedia Signal Processing (MMSP), 2010 IEEE International Workshop on, IEEE, 2010, pp. 350-355.
[14] B. Gao, C. Hao, T. Li, J. Ye, Grid generation on free-form surface using guide line advancing and surface flattening method, Advances in Engineering Software, 110 (2017) 98-109.
[15] C.-H. Peng, H. Pottmann, P. Wonka, Designing patterns using triangle-quad hybrid meshes, ACM Transactions on Graphics (TOG), 37 (2018) 107.
[16] H. Pottmann, C. Jiang, M. Höbinger, J. Wang, P. Bompas, J. Wallner, Cell packing structures, Computer-Aided Design, 60 (2015) 70-83.
[17] K. Shimada, D.C. Gossard, Bubble mesh: automated triangular meshing of non-manifold geometry by sphere packing, in: Proceedings of the third ACM symposium on Solid modeling and applications, ACM, 1995, pp. 409-419.
[18] K. Shimada, D.C. Gossard, Automatic triangular mesh generation of trimmed parametric surfaces for finite element analysis, Computer Aided Geometric Design, 15 (1998) 199-222.
[19] A. Zheleznyakova, S.T. Surzhikov, Molecular dynamics-based unstructured grid generation method for aerodynamic applications, Computer Physics Communications, 184 (2013) 2711-2727.
[20] A. Zheleznyakova, Molecular dynamics-based triangulation algorithm of free-form parametric surfaces for computer-aided engineering, Computer Physics Communications, 190 (2015) 1-14.
[21] L. Piegl, W. Tiller, The NURBS book, Springer Science & Business Media, 2012.
[22] R. McNeel, Grasshopper generative modeling for Rhino, Computer software (2011b), http://www. grasshopper3d. com, (2010).
[23] M.d. Berg, O. Cheong, M.v. Kreveld, M. Overmars, Computational geometry: algorithms and applications, Springer-Verlag TELOS, 2008.
[24] Q. Du, M.D. Gunzburger, L. Ju, Constrained centroidal Voronoi tessellations for surfaces, SIAM Journal on Scientific Computing, 24 (2003) 1488-1506.
[25] Q. Du, V. Faber, M. Gunzburger, Centroidal Voronoi tessellations: Applications and algorithms, SIAM review, 41 (1999) 637-676.
[26] Y. Li, E. Zhang, Y. Kobayashi, P. Wonka, Editing operations for irregular vertices in triangle meshes, ACM Transactions on Graphics (TOG), 29 (2010) 153.
[27] W.H. Frey, D.A. Field, Mesh relaxation: a new technique for improving triangulations, International Journal for

Numerical Methods in Engineering, 31 (1991) 1121-1133.
[28] V. Surazhsky, C. Gotsman, Explicit surface remeshing, in: Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing, Eurographics Association, 2003, pp. 20-30.
[29] D.A. Field, Qualitative measures for initial meshes, International Journal for Numerical Methods in Engineering, 47 (2000) 887-906.
[30] Q. Wang, B. Gao, H. Wu, Triangular mesh generation on free-form surfaces based on bubble dynamics simulation, Engineering Computations, 36 (2019) 646-663.